ACKNOWLEDGEMENT

We are greatly indebted to our mini project guide Ms.Nitu L Pariyal for her able guidance throughout this work. It has been an altogether different experience to work with her and we would like to thank her for her help, suggestions and numerous discussions.

We gladly take this opportunity to thank Dr.Mrs. Rajurkar A. M. (Head of Computer Science & Engineering, MGM' s College of Engineering, Nanded).

We are heartily thankful to Dr. Mrs. Lathkar G. S. (Director, MGM's College of Engineering, Nanded) for providing facility during progress of mini project in java, also for her kindly help, guidance and inspiration. Last but not least we are also thankful to all those who help directly or indirectly to develop this mini project and complete it successfully.

With Deep Reverence,
Vedika Joshi [171]

Abstract

The goal of this mini-project is to develop a simple, console-based **Weather Data Analysis System** using Java. The system is designed to efficiently manage and analyze a monthly (30-day) temperature dataset. Using core Java concepts like arrays, loops, and conditional statements, the system allows the user to input daily temperatures and perform key analyses, including calculating the **average temperature**, and identifying the **hottest** and **coldest** days in the dataset. This project provides a fundamental, accurate, and cost-effective method for basic data analysis, demonstrating essential programming logic.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ACKNOWLEDGEMENT	01
	Abstract	02
	Table of Content	03
1	Problem Statement	04
2	Code of Weather Data Analysis	07
3	Output	10
4	Working Procedure	13
5	Conclusion	14

1.Problem Statement

Analyzing large sets of daily weather data manually is prone to errors and time-consuming, especially when calculating statistical values like averages or identifying extremes. The lack of an automated tool makes it challenging for quick, reliable environmental data assessment.

The problem addressed is: How to design and implement a simple, console-based **Weather Data Analysis System** using core Java concepts to digitally store, manage, and process 30 days of temperature data, enabling the accurate calculation of the monthly average and the efficient identification of the maximum and minimum temperature days.

1. Objective of the Project

- To develop a simple Weather Data Analysis System using Java.
- To allow the user to store and manage 30 days of daily temperature data using arrays.
- To accurately calculate the **average temperature** for the entire 30-day period.
- To enable efficient identification and display of the **hottest day** (maximum temperature).
- To enable efficient identification and display of the **coldest day** (minimum temperature).
- To implement a **menu-driven interface** for ease of operation.

2. System Flow

The System Flow outlines the sequential steps taken by the program during execution, often shown via a Flowchart.

1. **Start & Initialize:** Program begins; temperatures and days arrays are initialized.
2. **Input:** User enters 30 daily temperatures, which are stored in the {temperatures} array.
3. **Display Menu:** The interactive menu is presented to the user.
4. **Process Choice:** User enters a choice (1-4, or 0).
5. **Execute Function:**
    - **Display:** Loop iterates and prints all 30 temperature records.
    - **Average:** Summation loop calculates the total, and the total is divided by 30.
    - **Hottest/Coldest:** Linear search loop finds the maximum/minimum value and its corresponding index (day number).
6. **Loop/End:** If the choice is 0, the program terminates; otherwise, the menu reappears.

3.Program Variables and Their Purpose

VARIABLE	PURPOSE	DATA TYPE
sc	Accepts user input from the console.	Scanner
temperatures[]	Stores the daily temperature data for 30 days.	double[]
days[]	Stores the short forms for the days of the week.	String[]
choice	Stores the user's menu selection.	int
sum	Accumulator variable for calculating the total temperature.	double

4.1. Class Diagram

The project is implemented using a single main Java class, WeatherDataAnalysis , which contains the main method and four static helper methods to manage the operations.

Class: WeatherDataAnalysis
Attributes:
- temperatures: double[30]
- days: String[7]
- sc: Scanner

Class: WeatherDataAnalysis	
Methods:	
+ main(String[] args): void	
+ displayTemperatures(double[], String[]): void	
+ calculateAverage(double[]): void	
+ findHottestDay(double[]): void	
+ findColdestDay(double[]): void	

5. Program Structure

The code is structured into one main class with dedicated methods for modularity and clarity:

Module	Function	Core Java Construct
Main Method	Initialization and Menu Control	switch-case, while loop
Input	Accepts 30 daily temperatures	Scanner ,for loop
Display Module	Prints all temperatures	for loop
Processing Logic	Calculate Average, Hottest, Coldest	for loop, if statements, arithmetic operators

Table 9.2 Program Structure

2.CODE OF WEATHER DATA ANALYSIS

Java Program :

```
import java.util.Scanner;

public class WeatherDataAnalysis {

public static void main(String[] args) {
Scanner sc = new Scanner(System.in);

    // Array to store temperatures for 30 days
    double[] temperatures = new double[30];
    String[] days = {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};

    System.out.println("=== WEATHER DATA ANALYSIS SYSTEM ===\n");

    // Input temperatures
    System.out.println("Enter daily temperatures (in Celsius) for 30 days:");
    for (int i = 0; i < 30; i++) {
      System.out.print("Day " + (i + 1) + " (" + days[i % 7] + "): ");
      temperatures[i] = sc.nextDouble();
    }

    // Menu-driven system
    while (true) {
      System.out.println("\n======= MENU =======");
      System.out.println("1. Display All Temperatures");
      System.out.println("2. Calculate Average Temperature");
      System.out.println("3. Find Hottest Day");
      System.out.println("4. Find Coldest Day");
      System.out.println("0. Exit");
      System.out.print("\nEnter your choice: ");

      int choice = sc.nextInt();

      switch (choice) {
        case 1:
          displayTemperatures(temperatures, days);
          break;
        case 2:
          calculateAverage(temperatures);
          break;
        case 3:
          findHottestDay(temperatures);
          break;
        case 4:
          findColdestDay(temperatures);
```

```
                break;
            case 0:
                System.out.println("\nThank you for using Weather Analysis System!");
                sc.close();
                return;
            default:
                System.out.println("Invalid choice! Please try again.");
        }
    }
}

// Display all temperatures
public static void displayTemperatures(double[] temps, String[] days) {
    System.out.println("\n--- All Temperature Records ---");
    for (int i = 0; i < temps.length; i++) {
        System.out.printf("Day %2d (%s): %.1f  C\n", (i + 1), days[i % 7], temps[i]);
    }
}

// Calculate average temperature
public static void calculateAverage(double[] temps) {
    double sum = 0;
    for (double temp : temps) {
        sum += temp;
    }
    double avg = sum / temps.length;
    System.out.printf("\n--- Average Temperature ---\n");
    System.out.printf("Average: %.2f  C\n", avg);
}

// Find hottest day
public static void findHottestDay(double[] temps) {
    double max = temps[0];
    int maxDay = 0;

    for (int i = 1; i < temps.length; i++) {
        if (temps[i] > max) {
            max = temps[i];
            maxDay = i;
        }
    }

    System.out.println("\n--- Hottest Day ---");
    System.out.printf("Day %d with %.1f  C\n", (maxDay + 1), max);
}

// Find coldest day
public static void findColdestDay(double[] temps) {
    double min = temps[0];
    int minDay = 0;
```

```
        for (int i = 1; i < temps.length; i++) {
            if (temps[i] < min) {
                min = temps[i];
                minDay = i;
            }
        }

        System.out.println("\n--- Coldest Day ---");
        System.out.printf("Day %d with %.1f  C\n", (minDay + 1), min);
    }
}
```

3.OUTPUT

```
=== WEATHER DATA ANALYSIS SYSTEM ===

Enter daily temperatures (in Celsius) for 30 days:
Day 1 (Mon): 23
Day 2 (Tue):  23
Day 3 (Wed): 23
Day 4 (Thu):  23
Day 5 (Fri):   23
Day 6 (Sat): 23
Day 7 (Sun): 2
Day 8 (Mon): 4
Day 9 (Tue): 24
Day 10 (Wed): 24
Day 11 (Thu): 24
Day 12 (Fri): 24
Day 13 (Sat): 24
Day 14 (Sun): 24
Day 15 (Mon): 24
Day 16 (Tue): 24
Day 17 (Wed): 24
Day 18 (Thu): 21
Day 19 (Fri): 34
Day 20 (Sat): 32
Day 21 (Sun): 32
Day 22 (Mon): 32
Day 23 (Tue): 34
Day 24 (Wed): 32
Day 25 (Thu): 33
Day 26 (Fri): 34
Day 27 (Sat): 35
Day 28 (Sun): 34
Day 29 (Mon): 36
Day 30 (Tue): 43

======= MENU =======
1. Display All Temperatures
2. Calculate Average Temperature
3. Find Hottest Day
4. Find Coldest Day
0. Exit

Enter your choice: 1

--- All Temperature Records ---
Day  1 (Mon): 23.0 C
Day  2 (Tue): 23.0 C
Day  3 (Wed): 23.0 C
Day  4 (Thu): 23.0 C
Day  5 (Fri): 23.0 C
Day  6 (Sat): 23.0 C
Day  7 (Sun): 2.0 C
Day  8 (Mon): 4.0 C
Day  9 (Tue): 24.0 C
Day 10 (Wed): 24.0 C
Day 11 (Thu): 24.0 C
Day 12 (Fri): 24.0 C
Day 13 (Sat): 24.0 C
Day 14 (Sun): 24.0 C
Day 15 (Mon): 24.0 C
Day 16 (Tue): 24.0 C
Day 17 (Wed): 24.0 C
Day 18 (Thu): 21.0 C
Day 19 (Fri): 34.0 C
Day 20 (Sat): 32.0 C
Day 21 (Sun): 32.0 C
Day 22 (Mon): 32.0 C
Day 23 (Tue): 34.0 C
Day 24 (Wed): 32.0 C
Day 25 (Thu): 33.0 C
Day 26 (Fri): 34.0 C
Day 27 (Sat): 35.0 C
Day 28 (Sun): 34.0 C
Day 29 (Mon): 36.0 C
Day 30 (Tue): 43.0 C

======= MENU =======
1. Display All Temperatures
2. Calculate Average Temperature
3. Find Hottest Day
4. Find Coldest Day
0. Exit

Enter your choice: 2

--- Average Temperature ---
Average: 26.40 C

======= MENU =======
1. Display All Temperatures
2. Calculate Average Temperature
3. Find Hottest Day
4. Find Coldest Day
0. Exit

Enter your choice: 3

--- Hottest Day ---
```

```
Day 30 with 43.0 C

======= MENU =======
1. Display All Temperatures
2. Calculate Average Temperature
3. Find Hottest Day
4. Find Coldest Day
0. Exit

Enter your choice: 4

--- Coldest Day ---
Day 7 with 2.0 C

======= MENU =======
1. Display All Temperatures
2. Calculate Average Temperature
3. Find Hottest Day
4. Find Coldest Day
0. Exit

Enter your choice: 0

Thank you for using Weather Analysis System!
```

4.Working Procedure

The user interacts with the system via the console:

1. **Initialization:** The program starts and prompts the user for input.
2. **Data Entry:** The user enters 30 daily temperatures sequentially.
3. **Menu Selection:** The menu is displayed, and the user selects an option (1, 2, 3, 4, or 0).
4. **Data Processing:**
   - If **Display**, all 30 days are printed.
   - If **Average**, the system sums the array and divides by 30 to display the average.
   - If **Hottest/Coldest**, the system iterates through the array, tracking the current extreme value and its index (day number).
5. **Termination:** The process repeats until the user selects option 0 (Exit).

CONCLUSION

The **Weather Data Analysis System** successfully achieved all project objectives. It is a functional and reliable Java application that automates the tedious processes of weather data processing, significantly enhancing efficiency and accuracy compared to manual methods. The effective use of arrays for data storage and basic algorithms for statistical analysis proves the ability of fundamental programming constructs to create practical solutions. The project confirms the efficiency of digital data management and serves as an excellent demonstration of applied Java programming.